|  |  |
| --- | --- |
| Name | Registration |
| Actors | Guest |
| Entry conditions | Null |
| Event flow | 1. Guest clicks on “Register” button on the homepage, either via mobile app or web app. 2. Guest fills the registration form with at least the mandatory fields. 3. Guest clicks on “Submit” button. 4. The system verifies the guest’s inputs 5. The system creates the new customer’s account and sends a confirmation to the new registered user. |
| Output conditions | Registration done and new customer added to the system. |
| Exceptions | 1. Guest does not fill correctly the fields. 2. Guest is already a user and the account already exists. 3. Mail, username or phone number are not valid or already taken by another user. |

|  |  |
| --- | --- |
| Name | Log in |
| Actors | Registered user |
| Entry conditions | User is already registered into the system. |
| Event flow | 1. User clicks on “log in” button on the homepage. 2. User fills the fields of the log in form. 3. The system verifies the fields filled by the user. 4. The system shows the user his user page. |
| Output conditions | Registered users are granted access to their personal page. |
| Exceptions | The user does not fill correctly any field of the log in form and receives another log in form. |

|  |  |
| --- | --- |
| Name | Request a ride |
| Actors | Customer, Taxi Driver |
| Entry conditions | Customer and Taxi Driver are registered and logged into the system. |
| Event flow | 1. Customer clicks on “Request a ride” button on his user page. 2. Customer fills the input form with at least the Origin field which is mandatory. 3. Customer clicks on “submit” button. 4. The system verifies the validity of the input form. 5. The system checks if the request overlaps with other active requests or reservations. 6. The system sends a confirmation to the customer. 7. The system stars searching for a taxi driver available to fullfill the request. 8. The system finds such a taxi driver and sends him the request. 9. The taxi driver confirms his intention to take care of the request, his status is set to “Busy”. 10. The system sends the waiting time and the taxi code to the user. |
| Output conditions | The customer successfully receives the taxi ride which he asked for. |
| Exceptions | 1. Customer does not fill correctly the request form. 2. Customer’s request overlaps with others active requests or reservations. |

|  |  |
| --- | --- |
| Name | Reserve a ride |
| Actors | Customer, Taxi Driver |
| Entry conditions | Customer and Taxi Driver are registered and logged into the system. |
| Event flow | 1. Customer clicks on “reserve a ride” button on his user page. 2. Customer fills the input form. All the fields are mandatory. 3. Customer clicks on “submit” button. 4. The system verifies the validity of the input form 5. The system sends a confirmation to the customer. 6. 10 minutes before the established time the system checks for overlaps with other active requests or reservations. 7. The system starts searching for a taxi available to take care of the customer. 8. The system finds a taxi and asks his confirmation. 9. The taxi driver confirms his intention to take care of the customer. His status is set to “Busy”. 10. The system sends the taxi code to the customer. |
| Output conditions | The customer successfully receives the taxi ride he reserved. |
| Exceptions | 1. Customer does not fill correctly the reservation form. 2. Customer reservation overlaps with other active requests or reservations. |

|  |  |
| --- | --- |
| Name | Delete a ride |
| Actors | Customer |
| Entry conditions | 1. Customer is already registered and logged into the application. 2. There is an actual request or reservation to delete. |
| Event flow | 1. Customer clicks on “History” button on his user page. 2. Customer selects a request or a reservation from the list of his unaccomplished taxi rides. 3. Customer clicks on “info” button. 4. Customer clicks on “delete” button. 5. System checks that the customer is allowed to delete the ride. 6. The system sends a notification to communicate either the success or fail of the operation. |
| Output conditions | The customer successfully delete a request or reservation previously made, there is no need to fullfill it anymore. |
| Exceptions | A taxi has already been assigned to the request/reservation, the customer cannot delete it. |

|  |  |
| --- | --- |
| Name | Set availability |
| Actors | Taxi Driver |
| Entry conditions | 1. The taxi driver is already registered and logged into the application. 2. His status is either “Available” or “Busy”. |
| Event flow | 1. Taxi driver clicks either on “set available” or “set busy” button via his mobile application. 2. If the new status is “available” the system checks for any ride associated to the taxi and stores it among the completed rides. 3. If the new status is “available”, the system retrieves the taxi location via the GPS system. 4. The system updates a taxi queue either removing or inserting the taxi, depending on the new status selected. |
| Output conditions | The taxi driver correctly change his status and the system reacts according to it by updating his taxi queues. |
| Exceptions | Null |

|  |  |
| --- | --- |
| Name | Create/Delete taxi driver’s account. |
| Actors | Administrator |
| Entry conditions | 1. If “delete”, there must be an account to delete. 2. The administrator must be logged into the system. |
| Event flow | 1. Administrator clicks on “create account” / “delete account” button. 2. The administrator fills a form with data about the account to be created/deleted. 3. Administrator clicks on “submit” button. 4. The system checks the validity of the operation. 5. The system creates/delete the specified account. |
| Output conditions | Taxi account created/deleted. |
| Exceptions | 1. Administrator does not fill correctly the form. 2. No such an account to be deleted exists. 3. There is already the account to be created. |

|  |  |
| --- | --- |
| Name | History |
| Actors | Customer |
| Entry Conditions | Customer already registered and logged into the application. |
| Event flow | 1. Customer clicks on “history” button. 2. Application shows the history of personal rides. 3. Customer may want to select a specific ride and see the information related. |
| Output conditions | The application allow the customer to see his personal history of taxi rides. |
| Exceptions | Null |

|  |  |
| --- | --- |
| Name | Log out |
| Actors | Registered user |
| Entry conditions | 1. User logged in. |
| Event flow | 1. User clicks on log out button. 2. If TaxiDriver, the system checks that his current status is not “Busy”. 3. If TaxiDriver, his status is set to “NotInService”. |
| Output conditions | User is no longer logged in the application and, if TaxiDriver, his current status becomes “NotInService”. |
| Exceptions | User is TaxiDriver and his current status is “Busy”. |

|  |  |
| --- | --- |
| Name | ModifyRideStatus |
| Actors | Administrator |
| Entry conditions | Null |
| Event flow | 1. Admin selects a ride stored in the system. 2. Admin modifies the ride’s status. |
| Output condition | Ride’s status successfully modified. |
| Exceptions | Null |

|  |  |
| --- | --- |
| Name | ForceLogOut |
| Actors | Administrator |
| Entry conditions | A taxi driver is logged in and must be manually logged out. |
| Event flow | 1. Admin selects a TaxiDriver. 2. Admin manually forces the log out of the TaxiDriver. 3. TaxiDrivers status is set to NotInService and any active ride related to him is set to “Completed”. |
| Output conditions | TaxiDriver is logged out and any ride related to him is managed correctly. |
| Exception | Null |